**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA**   
  
Java Persistence API (JPA)

* JSR 338 Specification for persisting, reading and managing data from Java objects
* Does not contain concrete implementation of the specification
* Hibernate is one of the implementation of JPA

Hibernate

* ORM Tool that implements JPA

Spring Data JPA

* Does not have JPA implementation, but reduces boiler plate code
* This is another level of abstraction over JPA implementation provider like Hibernate
* Manages transactions

**Refer code snippets below on how the code compares between Hibernate and Spring Data JPA  
Hibernate**

   /\* Method to CREATE an employee in the database \*/

   public Integer addEmployee(Employee employee){

      Session session = factory.openSession();

      Transaction tx = null;

      Integer employeeID = null;

      try {

         tx = session.beginTransaction();

         employeeID = (Integer) session.save(employee);

         tx.commit();

      } catch (HibernateException e) {

         if (tx != null) tx.rollback();

         e.printStackTrace();

      } finally {

         session.close();

      }

      return employeeID;

   }

**Spring Data JPA**  
EmployeeRespository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

@Autowire

  private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

  employeeRepository.save(employee);

  }

​​​​​​​   
  
**Reference Links:**   
<https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1>   
<https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>

### 🔹 ****Java Persistence API (JPA)****

**Specification only** – it’s a standard (JSR 338) for ORM in Java.

**Defines interfaces** for entity management (EntityManager, Query, etc.).

**No implementation** – it needs a provider (like Hibernate).

Allows switching between providers easily.

**Example Providers: Hibernate, EclipseLink, OpenJPA**

### 🔹 ****Hibernate****

**Concrete implementation** of JPA.

ORM (Object-Relational Mapping) framework.

Provides **extra features** beyond JPA (caching, interceptors, etc.).

Can be used with or without JPA.

### 🔹 ****Spring Data JPA****

**Abstraction layer** built on top of JPA.

Works with any JPA provider (like Hibernate).

Reduces boilerplate by using **repositories**.

Integrates with **Spring Boot** for easy configuration.

Handles **transactions**, **query generation**, and **pagination** automatically.

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification | Implementation (of JPA) | Abstraction (on top of JPA) |
| Provides ORM? | No | Yes | Yes (via JPA provider like Hibernate) |
| Boilerplate Code | Moderate | High | Very Low |
| Implementation Needed | Yes (e.g., Hibernate) | Itself | Needs JPA Provider (like Hibernate) |
| Ease of Use | Medium | Medium | High |
| Spring Integration | Not Direct | Not Direct | Excellent |

